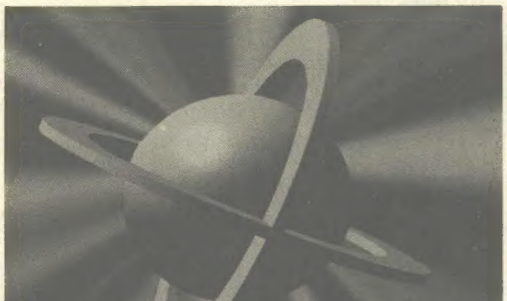
Unix

ไพโรจน์ ไววานิชกิจ

ก้าวแรก... สู่โลก UNIX ตอนที่ 4



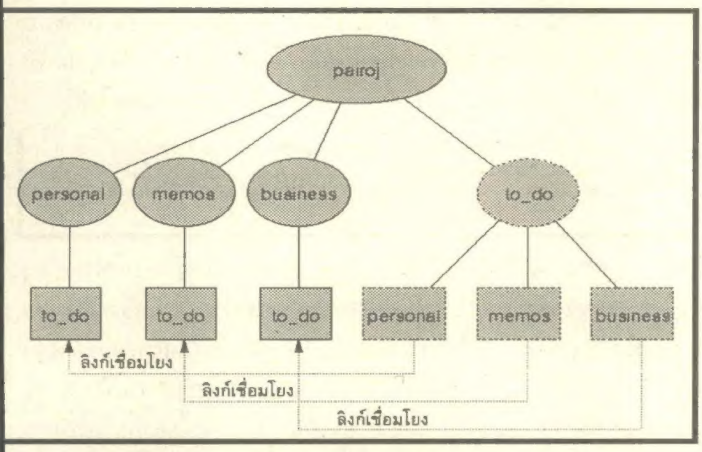
กล่าวต่อไปถึงเรื่องของการเชื่อมโยงไฟล์หรือลิงก์ ซึ่งเป็นวิธีในการดำเนินการจัดการกับไฟล์

จากบทความในตอนที่แล้ว ผู้เขียนได้กล่าวถึงการจัดโครงสร้างของระบบไฟล์และไดเรคทอรีของระบบปฏิบัติการ UNIX พร้อมกับได้ทิ้งท้ายไว้ด้วยเรื่องของการกำหนดสิทธิในการใช้ไฟล์ ในฉบับนี้จะได้กล่าวต่อไปถึงเรื่องของการเชื่อมโยงไฟล์หรือลิงก์ (link) ซึ่งเป็นวิธีการในการดำเนินการจัดการกับไฟล์ที่ได้รับความนิยมใช้งานเป็นอย่างมาก หลังจากนั้นจะได้กล่าวถึงเรื่องของการเชลล์และการใช้งานในรายละเอียดต่อไป

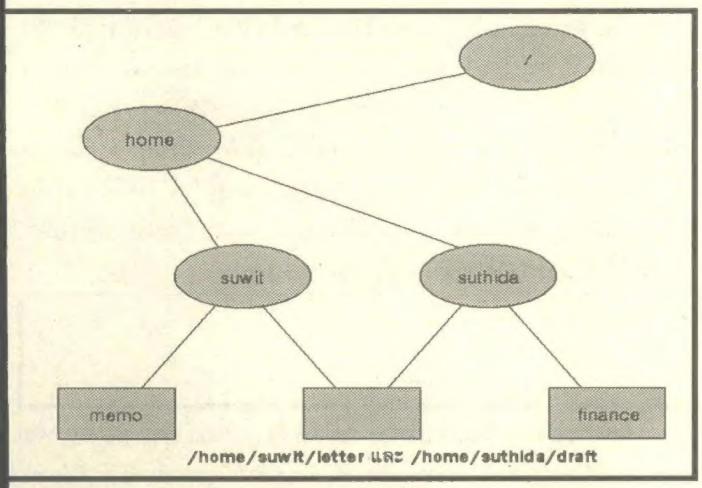
เรื่องของลิงก์

ความหมายของลิงก์ในระบบปฏิบัติการ UNIX ก็คือคำที่ใช้เรียกแทนพอยน์เตอร์ซึ่งชี้ระบุไปยังแต่ละไฟล์ทุกครั้งที่มีการสร้างไฟล์ใหม่ขึ้นในระบบไฟล์ไม่ว่าจะโดยใช้คำสั่ง **cat, vi, cp** หรือยูทิลิตี้อื่น ๆ UNIX จะทำการสร้างพอยน์เตอร์ขึ้นเพื่ออ้างอิงไฟล์ที่ถูกสร้างขึ้นนั้นเข้ากับพื้นที่ในฮาร์ดดิสก์ การดำเนินการใด ๆ กับไฟล์ดังกล่าวก็หมายถึงการดำเนินการกับพอยน์เตอร์ซึ่งอ้างอิงไปยังตำแหน่งที่เก็บไฟล์นั้นนั่นเอง ระบบปฏิบัติการ UNIX มีความสามารถในการเชื่อมโยงไฟล์ที่เป็นของผู้ใช้งานรายหนึ่งให้กับผู้ใช้งานรายอื่น ๆ ได้ร่วมกันใช้งาน โดยผู้ที่เป็นเจ้าของไฟล์จะต้องทำการกำหนดสิทธิในการใช้งานไฟล์นั้นโดยใช้คำสั่ง chmod ดังที่ได้กล่าวถึงในบทความตอนที่ 3 คำถามที่เกิดขึ้นตามมาก็คือ ประโยชน์ของการเชื่อมโยงไฟล์มีมากน้อยเพียงใด ผู้เขียนจะขอแสดงถึงด้วยการอ้างตามรูปที่ 1 ซึ่งเป็นการใช้ความสามารถในการเชื่อมโยงไฟล์กับการจัดการไฟล์ของผู้ใช้งานรายหนึ่ง

จากรูปจะเห็นว่าเป็นโครงสร้างภายใต้ไดเรคทอรี pairoj ซึ่งประกอบด้วยไดเรคทอรีรองจำนวน 3 ไดเรคทอรี (ไม่นับไดเรคทอรีที่แทนด้วยเส้นประ) ซึ่งได้แก่ personal, memos และ business แต่ละไดเรคทอรีจะมีไฟล์ข้อมูลที่มีชื่อว่า to_do ซึ่งเป็นไฟล์ที่ระบุเรื่องที่จะต้องดำเนินการในงานแต่ละอย่างตามชื่อไดเรคทอรีนั้น ในบางครั้งจะพบว่าการเปิดไฟล์ to_do ในแต่ละไดเรคทอรีขึ้นเพื่ออ้างอิงข้อมูลในเวลาเดียวกัน (เช่น การสรุปงานในเวลาเช้าหรือก่อนเลิกงาน) จะเป็นสิ่งที่ซึ่งค่อนข้างยุ่งยากในการอ้างชื่อไฟล์แต่ละไฟล์ แนวทางหนึ่งซึ่งช่วยลดความยุ่งยากก็คือการสร้างไดเรคทอรีใหม่ขึ้นภายในไดเรคทอรี pairoj มีชื่อว่า to_do (แสดงด้วยเส้นประ) โดยให้ไฟล์ที่อยู่ภายใต้ไดเรคทอรีดังกล่าวเป็นไฟล์ to_do ซึ่งอยู่ใต้ไดเรคทอรี personal, memos และ business แต่ใช้ชื่อไฟล์ให้ตรงตามประเภทของงาน ด้วยความสามารถในการสร้างลิงก์ของระบบปฏิบัติการ UNIX จึงสามารถอ้างชื่อไฟล์ที่อยู่ภายใต้ไดเรคทอรี to_do ให้สัมพันธ์กับไฟล์ในไดเรคทอรีทั้งสามที่ต้องการจะอ้างถึงได้ โดยที่ผู้ใช้งานไม่จำเป็นต้องทำการสร้างไฟล์ขึ้นใหม่แต่อย่างใด การแก้ไขไฟล์สามารถกระทำได้ทั้งกับไฟล์ที่ถูกอ้างถึงซึ่งอยู่ในไดเรคทอรีใดไดเรคทอรีหนึ่งทั้งสามหรือกับไฟล์ภายใต้ไดเรคทอรี to_do



รูปที่ 1 การจัดรูปแบบการอ้างอิงไฟล์โดยใช้การสร้างลิงก์



รูปที่ 2 ตัวอย่างการสร้างลิงก์เชื่อมโยงไฟล์ข้อมูล

การเชื่อมโยงไฟล์ในความหมายของระบบปฏิบัติการ UNIX ก็คือการสร้างลิงก์เพิ่มขึ้นนอกเหนือจากลิงก์ที่เชื่อมโยงไฟล์ไปยังฮาร์ดดิสก์ ลิงก์ที่ถูกสร้างขึ้นใหม่นี้จะเป็นการเชื่อมโยงไฟล์ที่ถูกอ้างถึงเข้ากับชื่อไฟล์ใหม่อีกชื่อหนึ่ง ในกรณีที่ไฟล์ที่ถูกอ้างถึงและไฟล์ที่ต้องการสร้างขึ้นใหม่อยู่ภายใต้ไดเรคทอรีเดียวกัน ผู้ใช้งานจะต้องระวังไม่ตั้งชื่อไฟล์ใหม่ให้ตรงกับชื่อไฟล์ที่ต้องการอ้างถึง ยกเว้นในกรณีที่ไฟล์ทั้งสองอยู่ต่างไดเรคทอรีกัน รูปที่ 2 เป็นตัวอย่างการเชื่อมโยงไฟล์ที่อยู่ภายใต้ไดเรคทอรี suwit มีชื่อไฟล์ว่า /home/suwit/letter ให้ไปปรากฏในไดเรคทอรี suthida ภายใต้ชื่อว่า /home/suthida/draft ซึ่งผู้ใช้งานที่ชื่อว่า suwit จะต้องทำการกำหนดสิทธิในการเข้าถึงไฟล์ให้กับ suthida โดยคำสั่ง chmod ตามความเหมาะสมว่าจะให้อ่านหรือเขียนอย่างเดียวหรือทั้งอ่านและเขียน ทั้งนี้ผู้ใช้งาน suthida จะเป็นผู้ใช้คำสั่ง **ln** สำหรับสร้างลิงก์เชื่อมโยงดังต่อไปนี้

```
$ pwd
/home/suthida
$ ln draft /home/suwit/letter
```

คำสั่ง **ln** จะทำหน้าที่สร้างพอยน์เตอร์เพิ่มเติมชี้ไปยังไฟล์ /home/suwit/letter ทั้งนี้พึงเข้าใจว่าการใช้คำสั่งดังกล่าวมิใช่เป็นการสำเนาไฟล์จาก /home/suwit/letter ไปเป็น /home/suthida/draft แต่อย่างใด ไฟล์ข้อมูลจริง ๆ ก็ยังคงมีอยู่เพียงไฟล์เดียวเหมือนดังก่อนหน้าจะมีการใช้คำสั่ง และตำแหน่งที่อยู่ของไฟล์ก็ยังคงเป็นภายใต้ไดเรคทอรี /home/suwit ข้อสังเกตอีกประการหนึ่งก็คือคุณสมบัติต่าง ๆ ของไฟล์ใหม่ เช่น สิทธิการจัดการไฟล์ เจ้าของไฟล์ รวมไปถึงวันเวลาที่ไฟล์ดังกล่าวถูกสร้างขึ้นก็ยังคงเป็นค่าเดียวกันกับไฟล์ที่ถูกอ้างถึงทุกประการ มีเพียงสิ่งที่อาจจะต่างกัน ก็คือ ชื่อของไฟล์ (ในกรณีที่ลิงก์โดยเปลี่ยนชื่อไฟล์ใหม่) ผู้อ่านสามารถทำการตรวจสอบเพื่อยืนยันว่าคำสั่ง **ln** มิได้เป็นคำสั่งที่ใช้สำเนาไฟล์โดยการป้อนชุดคำสั่งเพื่อสร้างลิงก์เชื่อมโยงไปยังไฟล์ ๆ หนึ่งตามต้องการ จากนั้นจึงทำการแก้ไขเนื้อความของไฟล์ใหม่ ทำการบันทึกข้อความแล้วลองเปิดอ่านไฟล์ที่ถูกอ้างถึง จะพบว่าเนื้อความของไฟล์ที่ถูกอ้างถึงได้เปลี่ยนแปลงไปตามที่ได้แก้ไขไว้ในไฟล์ใหม่ ผู้อ่านที่มิได้ใช้งานระบบปฏิบัติการ UNIX สามารถศึกษาได้จากตัวอย่างในกรอบต่อไปนี้

```
$ cat file_A
This is file A.
$ ln file_A file_B
$ cat file_B
This is file A.
$ vi file_B
This is the file B after change.<-- ข้อความที่พิมพ์แก้ไขเข้าไปใหม่
$ cat file_B
This is the file B after change.
$ cat file_A
This is the file B after change.<-- เนื้อความในไฟล์ file_A เปลี่ยนไปตามที่แก้ไขไว้
```

เมื่อเปรียบเทียบกับการสร้างไฟล์ขึ้นใหม่ด้วยวิธีสำเนาโดยการใช้คำสั่ง cp จะพบว่าการแก้ไขเนื้อความของไฟล์ใหม่จะส่งผลกระทบต่อการเปลี่ยนแปลงใด ๆ กับไฟล์ต้นฉบับแต่อย่างใด วิธีการหนึ่งที่ผู้ใช้งานสามารถนำไปใช้ในทางปฏิบัติเพื่อบอกได้ว่าไฟล์ใดเป็นฝ่ายที่เกิดจากการลิงก์ก็คือการใช้คำสั่ง **ls -l** ซึ่งจะเป็นการแสดงรายละเอียดของไฟล์ที่ต้องการตรวจสอบ โดยขอให้ดูจากกรอบตัวอย่างข้างล่างอันเป็นผลที่ได้จากการใช้คำสั่ง **ls -l** ภายใต้ไดเรคทอรีหนึ่งซึ่งประกอบด้วยไฟล์ข้อมูลจำนวน 4 ไฟล์ โดยที่มีไฟล์จำนวนหนึ่งไฟล์เกิดขึ้นจากการลิงก์หรือเชื่อมโยงกับอีกไฟล์หนึ่ง

```
$ ls -l report1 report2 report3 report4
-rw-r-r- 2 pairoj 33 Nov 5 11:55 report1
-rw-r-r- 2 pairoj 33 Nov 5 11:55 report2
-rw-r-r- 1 pairoj 16 Nov 5 11:25 report3
-rw-r-r- 1 pairoj 32 Nov 5 13:11 report4
```

จะเห็นได้ว่าไฟล์ report1 และ report2 เป็นไฟล์ที่ถูกลิงก์เข้าหากัน โดยดูจากจำนวนของลิงก์เชื่อมโยงซึ่งเป็นข้อมูลที่แสดงอยู่หลังจากสิทธิการใช้ไฟล์ (-rw-r—r—) โดยไฟล์ report1 มีจำนวนลิงก์ทั้งสิ้น 2 ลิงก์อันหมายถึงจำนวนพอยน์เตอร์ซึ่งชี้จากไฟล์ดังกล่าวไปยังพื้นที่เก็บข้อมูลในฮาร์ดดิสก์ 1 ลิงก์และพอยน์เตอร์ที่ลิงก์ไปยังไฟล์ report2 อีก

จำนวน 1 ลิงก์เช่นเดียวกันกับไฟล์ report2 สำหรับไฟล์ report3 และ report4 มีจำนวนลิงก์เพียง 1 ลิงก์เท่านั้น ซึ่งแสดงว่ามิได้เกิดจากการเชื่อมโยงไฟล์หรือมิได้เป็นต้นฉบับสำหรับการเชื่อมโยงโดยการลิงก์ไปยังไฟล์อื่น

ในกรณีที่ต้องการลบไฟล์ซึ่งถูกสร้างขึ้นด้วยวิธีการลิงก์ ผู้ใช้งานสามารถทำได้โดยการใช้คำสั่ง **rm** (remove) เช่นเดียวกับในกรณีที่ต้องการลบไฟล์ปกติที่มิได้เกิดจากลิงก์ รูปแบบของคำสั่งก็คือ **rm [ชื่อไฟล์]** หลังจากไฟล์ถูกลบไปแล้ว ผู้ใช้งานจะไม่สามารถทำการอ่านหรือแก้ไขไฟล์นั้น ๆ ได้อีกต่อไป นอกจากจะดำเนินการอ่านเขียนกับไฟล์ต้นฉบับเท่านั้น

สำหรับลิงก์เชื่อมโยงซึ่งผู้เขียนได้กล่าวถึงนี้มักเรียกกันว่า ฮาร์ดลิงก์ (hard link) นอกจากฮาร์ดลิงก์แล้ว ระบบปฏิบัติการ UNIX รุ่นใหม่ ๆ ส่วนใหญ่ได้รับการออกแบบให้สนับสนุนการสร้างลิงก์เชื่อมโยงแบบซิมโบลิคลิงก์ (symbolic link) เหตุผลที่มีการสร้างซิมโบลิคลิงก์ขึ้นก็เนื่องมาจากข้อจำกัดของฮาร์ดลิงก์เอง ทั้งนี้เนื่องจากในกรณีที่ต้องการสร้างฮาร์ดลิงก์เชื่อมโยงจากไดเรคทอรีหนึ่งไปเป็นไดเรคทอรีอีกชื่อหนึ่ง (อย่าลืมว่าไดเรคทอรีก็คือไฟล์ชนิดหนึ่ง) ผู้ที่สามารถทำการสร้างลิงก์ดังกล่าวได้ก็มีเพียงซูเปอร์ยูสเซอร์ (superuser) ซึ่งเป็นผู้ใช้งานที่มีสิทธิสูงสุดในระบบแต่เพียงผู้เดียวเท่านั้น ทำให้เกิดความลำบากต่อการสร้างฮาร์ดลิงก์ขึ้นในระดับไดเรคทอรีโดยตัวผู้ใช้งานทั่ว ๆ ไปเอง เปรียบเทียบกับการที่ผู้ใช้งานทั่วไปสามารถสร้างซิมโบลิคลิงก์เพื่อเชื่อมโยงได้ทั้งในระดับของไฟล์หรือในระดับของไดเรคทอรีได้อย่างเป็นอิสระ ยิ่งไปกว่านั้นผู้ใช้งานทุกคนก็ยังสามารถสร้างซิมโบลิคลิงก์เพื่อเชื่อมโยงระหว่างไฟล์ที่อยู่ต่างระบบไฟล์กันได้ ต่างกับฮาร์ดลิงก์ซึ่งจะเชื่อมโยงไฟล์ที่อยู่ในระบบไฟล์เดียวกันเท่านั้น (ผู้เขียนจะได้กล่าวถึงรายละเอียดของระบบไฟล์ในภายหลัง)

การสร้างซิมโบลิงก์จากไฟล์หนึ่งไปยังอีกไฟล์หนึ่งสามารถทำได้โดยใช้คำสั่ง **ln** ตามด้วย -s ตัวอย่างที่จะยกขึ้นมาต่อไปนี้เป็นการสร้างซิมโบลิคลิงก์จากไฟล์ memorandum ไปยังไฟล์ใหม่ที่ชื่อว่า /tmp/draft หลังจากนั้นจะเป็นการใช้คำสั่ง **ls -lg** เพื่อแสดงรายละเอียดของไฟล์ทั้งสอง จะเห็นได้ว่าในกลุ่มอักขระแสดงสิทธิการใช้ไฟล์ของไฟล์ /tmp/draft เริ่มต้นด้วยตัวอักษร l ซึ่งแสดงถึงการเป็นไฟล์ที่เกิดจากซิมโบลิคลิงก์แทนที่จะเป็นเครื่องหมาย - ดังเช่นในกรณีของไฟล์ทั่วไป นอกจากนั้นคุณสมบัติของไฟล์ทั้งสองยังมีความแตกต่างกันอีกด้วย สังเกตจากวันเวลาที่สร้างไฟล์ /tmp/draft ในส่วนหลังจากชื่อไฟล์ /tmp/draft ยังมีการแจ้งว่าไฟล์ดังกล่าวเกิดจากซิมโบลิคลิงก์ซึ่งชี้ไปยังไฟล์อ้างอิงคือ /tmp/memorandum ซึ่งผู้ใช้งานสามารถใช้คำสั่ง **ln -s** สำหรับสร้างซิมโบลิคจากไดเรคทอรีหนึ่งไปยังอีกไดเรคทอรีหนึ่งได้

```
$ ln -s memorandum /tmp/draft
$ ls -lg memorandum /tmp/draft
-rw-r--r-1 peter finance 981 Nov 6 10:55 memorandum
lrwxrwxrwx 1 peter finance 4 Nov 6 11:20 /tmp/draft ->
memorandum
```

ข้อสังเกตประการหนึ่งของการทำงานกับซิมโบลิคลิงก์ก็คือเมื่อใช้ร่วมกับคำสั่งบางคำสั่งของระบบปฏิบัติการ UNIX จะก่อให้เกิดความสับสนบางประการต่อตัวผู้ใช้งานเอง ลองพิจารณาตัวอย่างต่อไปนี้

```
$ ln -s /home/pairoj/saving /tmp/saving.old
$ cd /tmp/grades.old
$ pwd
/home/pairoj/saving
```

จะเห็นว่าคำสั่ง **pwd** ของ UNIX ไม่สามารถอ้างถึงตำแหน่งที่อยู่จริงของไดเรคทอรีใช้งานปัจจุบันที่เกิดจากการสร้างซิมโบลิคลิงก์ได้ โดยผลลัพธ์ที่คำสั่ง **pwd** แสดงออกมาจะเป็นตำแหน่งของไดเรคทอรีอ้างอิงแทน ซึ่งในการใช้งานจริงจะก่อให้เกิดความสับสนต่อผู้ใช้งานที่ไม่ทราบถึงข้อจำกัดนี้พอสมควร ในกรณีของเชลล์แบบซี (C Shell) ได้มีการกำหนดตัวแปรซึ่งมีชื่อว่า **cwd** (current working directory) ขึ้นเพื่อแสดงชื่อของซิมโบลิคลิงก์ของไดเรคทอรีที่ต้องการอ้างถึง หากใช้ตัวแปร **cwd** อ้างไปยังไดเรคทอรีที่มิได้เกิดจากซิมโบลิคลิงก์ก็จะได้ผลลัพธ์เป็นชื่อของฮาร์ดลิงก์ที่เชื่อมโยงไปยังไดเรคทอรีนั้น ๆ เนื่องจาก **cwd** เป็นเพียงตัวแปรมิได้เป็นชุดคำสั่ง การอ้างถึงจึงต้องนำหน้าด้วยเครื่องหมาย % และใช้คำสั่ง **echo** ในการบังคับให้แสดงค่า ผู้เขียนจะขอกล่าวถึงรายละเอียดของการควบคุมเชลล์ในบทความตอนต่อ ๆ ไป ในที่นี้ขอให้ดูตัวอย่างการใช้ตัวแปร **cwd** ดังต่อไปนี้ ขอให้สังเกตเครื่องหมายพรอมต์ซึ่งแสดงให้เห็นว่ากำลังทำงานอยู่ในเชลล์แบบซี

```
% pwd
/home/pairoj/saving
% echo $cwd
/tmp/saving.old
```

ข้อควรสังเกตอีกประการหนึ่งก็คือ หากมีการใช้คำสั่ง **cd** เพื่อย้ายตำแหน่งไดเรคทอรีทำงานแบบสัมพัทธ์จากไดเรคทอรีที่เกิดจากซิมโบลิค จะให้ผลเป็นการย้ายตำแหน่งจากไดเรคทอรีอ้างอิงแทน ตัวอย่างต่อไปนี้เป็นการใช้คำสั่ง **cd..** เพื่อย้ายตำแหน่งไดเรคทอรีทำงานขึ้นจากไดเรคทอรี /tmp/saving.old ซึ่งเป็นไดเรคทอรีทำงานปัจจุบัน หลังจากใช้คำสั่งดังกล่าวแล้ว ทำการตรวจสอบตำแหน่งไดเรคทอรีทำงานใหม่โดยใช้ตัวแปร **cwd** จะพบว่าไดเรคทอรีทำงานใหม่นั้นกลับกลายเป็น /home/pairoj แทนที่จะเป็น /tmp ผู้เขียนจึงใคร่ขอให้ผู้ใช้งาน UNIX ทุกท่านระมัดระวังและให้ความใส่ใจกับการทำงานกับไดเรคทอรีที่เกิดจากซิมโบลิคลิงก์ให้มาก เพื่อป้องกันความสับสนอันอาจจะเกิดขึ้นได้

```
% echo $cwd
/tmp/saving.old
% cd ..
% echo $cwd
/home/pairoj
```

สำหรับการลบไฟล์อ้างอิงซึ่งในที่นี้คือไฟล์ memorandum จะมีผลทำให้ซิมโบลิคลิงก์ซึ่งเชื่อมโยงไปยังไฟล์ /tmp/draft ถูกลบไปด้วยโดยทันที ผลที่ตามมาก็คือผู้ใช้งานไม่สามารถที่จะอ่านหรือแก้ไขเนื้อหาภายในไฟล์ /tmp/draft อีกต่อไป แต่ยังสามารถมองเห็นไฟล์ดังกล่าวได้เมื่อบังคับให้ระบบปฏิบัติการ UNIX แสดงผลโดยการใช้คำสั่ง **ls -lg**

ดังนั้นจึงควรระลึกไว้ว่าเมื่อทำการลบไฟล์อ้างอิงไปแล้ว ก็ควรที่จะทำการลบไฟล์ใหม่ที่เกิดจากซิมโบลิคลิงก์ด้วยการใช้คำสั่ง **rm**

```
$ ls -lg memorandum
-rw-r-r- 1 peter finance 981 Nov 6 10:55 memorandum
$ ln -s memorandum /tmp/draft
$ rm memorandum
$ cat /tmp/draft
/tmp/draft : No such file or directory
$ ls -lg /tmp/draft
lrwxrwxrwx 1 peter finance 4 Nov 6 11:20 /tmp/draft ->
memorandum
```

สำหรับเรื่องของการจัดการไฟล์และไดเรคทอรีของระบบปฏิบัติการ UNIX ก็คงจะมีเนื้อหาคร่าว ๆ เพียงเท่านี้ แต่ก่อนที่ผู้เขียนจะได้กล่าวต่อไปในเรื่องของเชลล์ ลองพิจารณาหาคำตอบจากคำถามต่อไปนี้

"สมมติว่าผู้อ่านได้สร้างไฟล์ ๆ หนึ่งซึ่งมีชื่อว่า andor ในไดเรคทอรีใช้งานของตนเอง หลังจากนั้นก็ได้พิมพ์คำสั่ง **mv andor andVor** ถามว่าจะเกิดอะไรขึ้น" ฝากไปคิดดูครับ

เรื่องของเชลล์

ผู้เขียนเข้าใจว่าผู้อ่านคงพอจะทราบคร่าว ๆ บ้างแล้วว่าเชลล์ในระบบปฏิบัติการ UNIX สำคัญและมีหน้าที่ประการใด ในตอนนี้จึงจะขอกล่าวถึงรายละเอียดพื้นฐานที่จำเป็นจะต้องทราบเกี่ยวกับเชลล์ อันได้แก่ รูปแบบการจัดชุดคำสั่ง กระบวนการในการจัดการกับชุดคำสั่งของเชลล์ อินพุตและเอาต์พุตมาตรฐาน การเปลี่ยนทิศทางอินพุตและเอาต์พุตของชุดคำสั่ง เรื่องของไปป์และฟิลเตอร์ รวมไปถึงเรื่องของการสั่งให้โปรแกรมทำงานแบบหลังฉาก ปิดท้ายด้วยเรื่องของการประมวลผลที่เกี่ยวกับไฟล์และลูกเล่นของการใช้อักขระพิเศษต่าง ๆ

หน้าที่ประการสำคัญของเชลล์ในระบบปฏิบัติการ UNIX คือการรับชุดคำสั่งที่ผู้ใช้งานป้อนผ่านพรอมต์ (prompt) ซึ่งอาจจะอยู่ในรูปของคำสั่งในระบบ UNIX เอง หรืออาจเป็นเชลล์สคริปต์ซึ่งจะได้กล่าวถึงในบทความตอนต่อ ๆ ไป หรือแม้กระทั่งเป็นโปรแกรมใช้งานต่าง ๆ ก่อนอื่นเราจะมาตกลงกันก่อนว่าบรรดาอักขระต่าง ๆ ที่ประกอบกันขึ้นเป็นคำสั่งซึ่งผู้ใช้งานป้อนผ่านพรอมต์ของระบบปฏิบัติการ UNIX จะอยู่ในบรรทัดที่มีชื่อเรียกว่าบรรทัดคำสั่ง หรือ command line ซึ่งโดยทั่วไปโครงสร้างของ command line จะอยู่ในรูปแบบต่อไปนี้

```
คำสั่ง [อาร์กิวเมนต์ 1] [อาร์กิวเมนต์ 2] ... [อาร์กิวเมนต์ n]
```

สำหรับอาร์กิวเมนต์ซึ่งเป็นตัวแปรตามหลังชุดคำสั่งนั้นอาจจะมีหรือไม่มี หรือมีเพียงไม่กี่ตัวก็ขึ้นอยู่กับว่าคำสั่งที่ป้อนและจุดประสงค์ใช้งานของผู้ใช้งานในขณะนั้นต้องการอ้างถึงอาร์กิวเมนต์กี่ตัว บางคำสั่งอาจไม่มีอาร์กิวเมนต์ตามเลยก็ได้ ผู้เขียนขอยกตัวอย่างคำสั่ง **ls** ซึ่งหากป้อนคำสั่งดังกล่าวโดยไม่มีพารามิเตอร์หรืออาร์กิวเมนต์ใด ๆ ตามหลังเลย ระบบปฏิบัติการ UNIX จะทำการแปลคำสั่งและแสดงรายชื่อไฟล์ที่อยู่ในไดเรคทอรีปัจจุบันโดยไม่แสดงรายละเอียดต่าง ๆ ของไฟล์มากเท่าเมื่อเทียบกับการป้อนคำสั่ง **ls -lg** อาร์กิวเมนต์เป็นได้ตั้งแต่ชื่อไฟล์ กลุ่มอักขระ หรือตัวเลข ขึ้นอยู่กับตัวคำสั่งเอง ยกตัวอย่าง เช่น คำสั่ง **vi** ต้องการอาร์กิวเมนต์ที่เป็นชื่อไฟล์ซึ่งต้องการอ่านหรือแก้ไข อีกตัวอย่างหนึ่งคือการใช้คำสั่ง **cp** ในการสำเนาไฟล์ที่ชื่อ temp ไปเป็นไฟล์ชื่อ temporary

```
$ cp temp temporary
```

จะเห็นว่าอาร์กิวเมนต์ทั้ง 2 ตัวเป็นชื่อไฟล์ที่เกี่ยวข้อง โดยเป็นชื่อไฟล์ต้นฉบับและชื่อไฟล์ใหม่ตามลำดับ ในกรณีนี้กล่าวได้ว่าจำเป็นต้องใช้อาร์กิวเมนต์ทั้ง 2 ตัว จะละตัวใดตัวหนึ่งไม่ได้ ซึ่งหากผู้ใช้งานบังเอิญป้อนอาร์กิวเมนต์ไม่ครบ ระบบปฏิบัติการ UNIX จะแจ้งเตือนกลับมาให้ผู้ใช้งานป้อนคำสั่งให้ถูกต้อง

นอกจากอาร์กิวเมนต์ประเภทที่บังคับให้ต้องป้อนแล้ว ยังมีอาร์กิวเมนต์อีกประเภทหนึ่งซึ่งผู้ใช้งานมีสิทธิที่จะป้อนหรือไม่ก็ได้ โดยหากป้อนลงไปจะเป็นการควบคุมรูปแบบการทำงานของโปรแกรมให้เป็นไปตามความหมายของอาร์กิวเมนต์นั้น ๆ ผู้ใช้งานสามารถป้อนอาร์กิวเมนต์ประเภทนี้ในคำสั่งเดียวกันได้มากกว่าหนึ่งตัว โดยทั่วไปจะเรียกอาร์กิวเมนต์ประเภทนี้ว่า ออปชัน (option) ระบบปฏิบัติการ UNIX หลายรายมักแยกความแตกต่างของออปชันออกจากอาร์กิวเมนต์ โดยการกำหนดให้มีเครื่องหมาย - นำหน้าออปชันทุกตัว ลองดูตัวอย่างต่อไปนี้

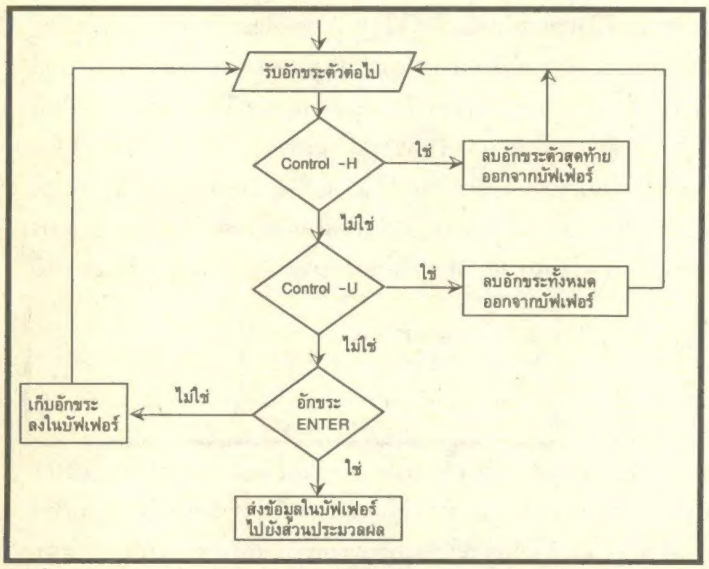
```
$ ls -r
test    personal names
temp    oldstuff        hold
$ ls -x
hold    names   oldstuff
personal        temp    test
```

จะเห็นว่าการใช้คำสั่ง **ls -r** (reverse order) ทำให้ระบบ UNIX ตอบสนองคำสั่ง **ls** โดยแสดงรายชื่อของไฟล์ในไดเรคทอรีปัจจุบันเรียงย้อนกลับตามลำดับอักขระ (บนลงล่างแล้วไล่จากซ้ายไปขวา) ส่วนออปชัน -x จะบังคับให้ UNIX แสดงรายชื่อไฟล์เรียงตามลำดับอักขระชื่อ เรียงตามแนวนอน ผู้เขียนขอแนะนำให้ผู้อ่านที่พึ่งเริ่มใช้งานระบบปฏิบัติการ UNIX ลองศึกษารายละเอียดของออปชันต่าง ๆ สำหรับคำสั่งพื้นฐานบางคำสั่ง ดังเช่น **ls** เป็นต้น อีกสิ่งหนึ่งที่จะขอกล่าวถึงก็คือผู้ใช้งานสามารถนำออปชันหลาย ๆ ตัวมารวมกันไว้ได้ใน command line เดียวกัน โดยพิมพ์อักขระที่ใช้แทนออปชันติดกันเป็นกลุ่ม นำหน้ากลุ่มด้วยเครื่องหมาย - อย่างไรก็ตาม คำสั่งหรือยูทิลิตี้บางตัวอาจไม่ยอมให้ผู้ใช้งานรวมออปชันต่าง ๆ เข้าด้วยกันตามอิสระ เป็นหน้าที่ที่ผู้ใช้งานต้องทำการศึกษาและทำความเข้าใจ ตัวอย่างของการรวมออปชันสำหรับคำสั่ง **ls** เป็นดังต่อไปนี้

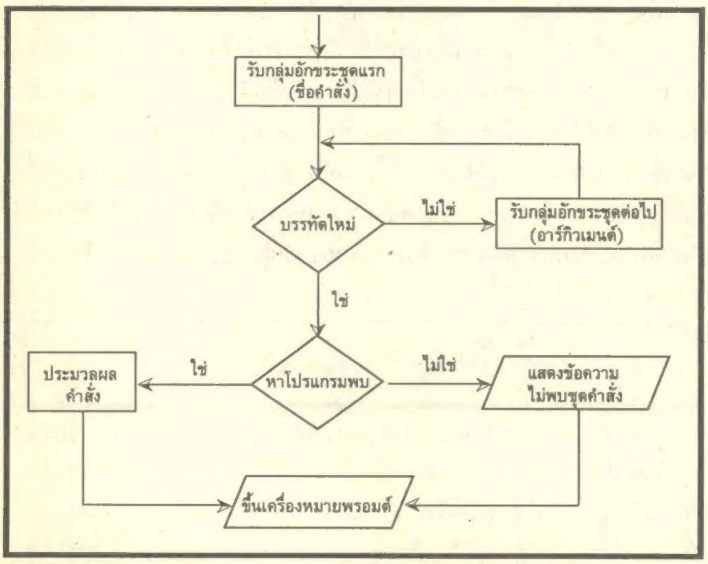
```
$ ls -rx
test    temp    personal
oldstuff        names   hole
```

กระบวนการแปลความคำสั่งของเชลล์ในระบบปฏิบัติการ UNIX แสดงในรูปที่ 3 ซึ่งจะเห็นว่าเชลล์รับอักขระที่ป้อนผ่านคีย์บอร์ดซึ่งเป็นอุปกรณ์อินพุตมาตรฐาน (หรือในบางครั้งอาจมาจากแหล่งอินพุตอื่น ๆ เช่นจากไฟล์ ดังจะได้กล่าวในเรื่องของการเปลี่ยนทิศทาง) มาทำการเก็บลงในบัฟเฟอร์ทีละอักขระจนกว่าจะพบอักขระสัญลักษณ์ ENTER ซึ่ง

เป็นการแสดงการสิ้นสุด command line จึงทำการส่งกลุ่มอักขระที่ได้รับทั้งหมดในบัฟเฟอร์ไปยังส่วนประมวลผลคำสั่ง (shell processing) ในกรณีที่ผู้ใช้งานป้อนอักขระควบคุม Control-H ซึ่งเป็นการสั่งลบอักขระที่ป้อนผ่านตัวล่าสุด 1 ตัว หรืออักขระควบคุม Control-U ซึ่งเป็นการสั่งลบข้อความที่ป้อนมาทั้งบรรทัด ส่วนควบคุมการรับข้อมูลของเชลล์จะตอบสนองกับคำสั่งดังกล่าวทันที โดยในกรณีของ Control-H เชลล์จะทำการลบอักขระตัวสุดท้ายที่ถูกเก็บลงในบัฟเฟอร์ออกแล้วรอรับอักขระที่จะถูกป้อนเข้ามาตัวต่อไป สำหรับกรณีของ Control-W เชลล์จะทำการลบอักขระทั้งหมดที่ถูกเก็บลงในบัฟเฟอร์พร้อมกับรอรับการป้อนคำสั่งชุดใหม่



รูปที่ 3 โฟลว์ชาร์ตการรับคำสั่งจากอินพุตมาตรฐานของระบบปฏิบัติการ UNIX



รูปที่ 4 โฟลว์ชาร์ตการประมวลชุดคำสั่ง

สำหรับการทำงานของส่วนประมวลผลคำสั่งจะเป็นไปตามแผนภาพในรูปที่ 4 เชลล์จะรับกลุ่มอักขระทั้งหมดที่ประกอบกันขึ้นเป็น command line ซึ่งส่งมาจากบัฟเฟอร์อินพุต จากนั้นจึงทำการแบ่ง command line ออกเป็นส่วนย่อย โดยให้เป็นส่วนของคำสั่ง และส่วนของอาร์กิวเมนต์ต่าง ๆ สิ่งที่ใช้ในการพิจารณาแยกส่วนประกอบของ command line ก็คือช่องว่าง (space) ระหว่างกลุ่มคำ ซึ่งอาจเกิดจากการพิมพ์เครื่องหมาย space หรือ TAB ก็ได้ ส่วนประมวลผลจะตรวจสอบว่าคำสั่งที่ป้อนเข้ามาคือคำสั่งอะไร โดยดูจากกลุ่มคำกลุ่มแรกไล่จากอักษรตัวแรกไปจนถึงอักษรตัวสุดท้าย สำหรับในส่วนของกลุ่มคำที่แทนถึงคำสั่งนั้นอาจจะอยู่ในรูปของชื่อคำสั่งหรืออยู่ในรูปของการอ้างคำสั่งแบบสัมบูรณ์ก็ได้ ส่วนประมวลจะมองว่าคำสั่ง **ls** หรือ **/bin/ls** เป็นคำสั่งเดียวกัน โดยหากเมื่อใดก็ตามที่พบว่ากลุ่มคำแรกของ command line ขึ้นต้นด้วยเครื่องหมาย / ส่วนประมวลผลจะอ้างอิงตำแหน่งของตัวคำสั่งหลังเครื่องหมาย / ตัวสุดท้ายในกลุ่มคำนั้น อย่างไรก็ตาม ในกรณีของการอ้างคำสั่งแบบสัมพัทธ์ เช่น **ls** นั้น ผู้ใช้งานจะต้องแน่ใจว่าได้ทำการติดตั้งชื่อ PATH ซึ่งเก็บโปรแกรมนั้นให้ระบบปฏิบัติการ UNIX ทราบล่วงหน้าก่อนแล้ว ผู้เขียนจะได้กล่าวถึงในเรื่องนี้ในบทความตอนต่อ ๆ ไป

เมื่อส่วนประมวลผลสามารถแปลความหมายของคำสั่งที่ถูกป้อนให้ได้แล้ว ก็จะตรวจสอบว่ามีอาร์กิวเมนต์ใดตามมาบ้าง โดยใช้เทคนิคในการตรวจสอบกลุ่มคำตามที่ได้กล่าวมาแล้วจนครบทุกอาร์กิวเมนต์ จากนั้นจึงทำการประมวลผลตามคำสั่งนั้น หากไม่พบคำสั่งดังกล่าวก็จะมีการแจ้งข้อความผิดพลาดผ่านทางอุปกรณ์เอาต์พุตมาตรฐานซึ่งโดยส่วนใหญ่จะเป็นจอภาพ เมื่อการประมวลผลสำหรับ command line ปัจจุบันสำเร็จลงแล้ว ส่วนประมวลผลจะรอรับ command line ชุดใหม่ที่ถูกส่งมาจากบัฟเฟอร์อินพุตเพื่อการประมวลผลต่อไป

อินพุตและเอาต์พุตมาตรฐาน

ผู้เขียนได้กล่าวถึงอินพุตและเอาต์พุตมาตรฐานสำหรับระบบปฏิบัติการ UNIX ในบทความตอนแรก ๆ มาบ้างแล้ว ในที่นี้จะขอกล่าวถึงรายละเอียดเพื่อให้เกิดความเข้าใจได้อย่างชัดเจนขึ้น เอาต์พุตมาตรฐานก็คือจุดที่เชลล์ทำการส่งผลลัพธ์ซึ่งเป็นข้อมูลข่าวสารต่าง ๆ ออกไป ข้อมูลเหล่านี้อาจจะถูกส่งออกไปยังเครื่องพิมพ์ ไฟล์ต่าง ๆ หรือออกไปยังเทอร์มินัลที่เชื่อมต่ออยู่ สำหรับอินพุตมาตรฐานก็คือจุดที่เชลล์จะรับข้อมูลข่าวสาร ซึ่งโดยทั่วไปก็ได้แก่คีย์บอร์ด แต่ระบบปฏิบัติการ UNIX สามารถใช้อุปกรณ์สื่อสารอื่น ๆ ซึ่งสามารถนำเข้าข้อมูลได้เป็นอินพุตมาตรฐานได้ ตัวอย่างเช่น การรับข้อมูลเข้าจากไฟล์ การรับข้อมูลเข้าจากพอร์ตสื่อสาร เป็นต้น

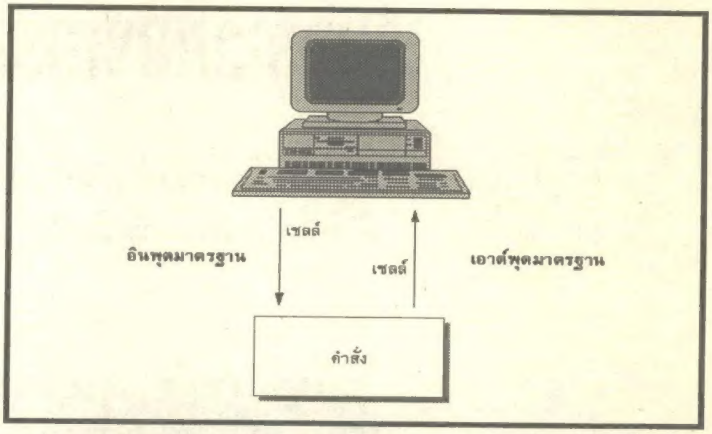
จากที่ได้กล่าวไว้ในบทความตอนที่แล้วว่าในระบบไฟล์ของระบบปฏิบัติการ UNIX มีไดเรคทอรีซึ่งมีชื่อว่า /dev ซึ่งเก็บไฟล์พิเศษประเภทหนึ่งที่มีชื่อว่าดีไวซ์ไฟล์ (device file) ดีไวซ์ไฟล์เป็นไฟล์ที่ UNIX ใช้ในการอ้างถึงแทนอุปกรณ์ต่อพ่วง เช่น เครื่องพิมพ์ อุปกรณ์

ื่อสารที่ต่อผ่านพอร์ตอนุกรม ฮาร์ดดิสก์ ฟลอปปีดิสก์ เป็นต้น ในกรณีผู้ใช้งานทำการ login เข้าสู่ระบบปฏิบัติการ UNIX แล้วลองใช้คำสั่ง who ตามด้วย login ของตนเอง จะเห็นว่าคำสั่งดังกล่าวมีการแสดงถึงชื่อของไวซ์ที่ตนใช้งานในการรับส่งข้อมูล (ชุดคำสั่ง) อยู่ สมมติว่ามีชื่อเป็น y02 การอ้างถึงดีไวซ์ดังกล่าวโดยระบบปฏิบัติการ UNIX ก็จะเป็นการดต่อไปยังไฟล์ /dev/tty02 ซึ่งโดยทั่วไปผู้ใช้งานทั่วไปอาจจะไม่สนใจากนักว่าตนเองได้รับการติดต่อปฏิสัมพันธ์จากระบบปฏิบัติการ UNIX านทางไฟล์ดังกล่าว การป้อนข้อมูลใด ๆ จากผู้ใช้งานผ่านทางอุปกรณ์ย์บอร์ดซึ่งถือเป็นส่วนหนึ่งของเทอร์มินัลก็คือการเขียนข้อมูลลงไปบนฟล์ /dev/tty02 ระบบ UNIX จะทำการอ่านข้อความที่ได้รับจากไฟล์ dev/tty02 เพื่อนำไปประมวลผลตามที่ได้กล่าวมาแล้ว และเมื่อมีการสดงผลตอบกลับมายังจอภาพ ระบบปฏิบัติการ UNIX ก็จะทำการเขียนอมูลลงบนไฟล์ /dev/tty02 ซึ่งก็จะเป็นสิ่งเดียวกับที่ปรากฏขึ้นบนจอภาพนเอง

โดยปกติระบบปฏิบัติการ UNIX จะตัดสินใจเลือกอุปกรณ์เอาต์พุตที่สอดคล้องกับอุปกรณ์อินพุตที่ผู้ใช้ทำการ login ผ่านเข้ามา ตัวอย่างเช่นในกรณีที่ผู้ใช้งานทำการ login เข้าสู่ระบบปฏิบัติการ UNIX โดยใช้เทอร์มินัลหรือคอนโซลซึ่งเชื่อมต่ออยู่กับเครื่องคอมพิวเตอร์หลักนพุตมาตรฐานที่ผู้ใช้งานทำการ login ผ่านเข้ามานั้นก็คือเทอร์มินัลย์บอร์ดเป็นส่วนหนึ่งของอุปกรณ์เทอร์มินัล) ในกรณีนี้ระบบปฏิบัติการNIX จะจัดให้อุปกรณ์เอาต์พุตมาตรฐานสำหรับติดต่อกับผู้ใช้งานรายนเป็นอุปกรณ์เทอร์มินัล (ใช้จอภาพเป็นเอาต์พุต) ซึ่งโครงสร้างการรูปแบบอินพุตและเอาต์พุตมาตรฐานแสดงอยู่ในรูปที่ 5

ตัวอย่างที่ผู้ใช้งานสามารถทดสอบดูเพื่อเพิ่มความเข้าใจในเรื่องงอุปกรณ์อินพุตและเอาต์พุตมาตรฐานก็คือการใช้คำสั่ง cat ดังเป็นที่ราบกันแล้วว่าการใช้คำสั่ง cat ตามด้วยชื่อไฟล์จะเป็นการสั่งให้ระบบฏิบัติการ UNIX ทำการแสดงเนื้อหาภายในไฟล์นั้นออกมาทันทีที่มีการอนคำสั่งพร้อมด้วยชื่อไฟล์ผ่านอุปกรณ์อินพุตซึ่งในที่นี้ก็คือคีย์บอร์ดลล์จะทำการตีความ command line ที่ได้รับ หลังจากนั้นก็จะทำการยนผลลัพธ์ลงบนดีไวซ์ไฟล์ซึ่งตรงกับอุปกรณ์เอาต์พุตมาตรฐาน ในณีนี้เอาต์พุตมาตรฐานคือจอภาพซึ่งเป็นส่วนหนึ่งของเทอร์มินัล ผู้ใช้สามารถมองเห็นสิ่งที่เชลล์ตอบกลับมาจากบนจอภาพของตนได้

คราวนี้ขอให้ลองพิมพ์คำสั่ง cat โดยไม่มีอาร์กิวเมนต์ใด ๆ ตามอมทั้งกดปุ่ม ENTER ในกรณีนี้เชลล์จะไม่พบไฟล์ที่จะใช้เป็นอินพุตตรฐานของคำสั่งนั้น (ต่างกับในกรณีแรกตรงที่อินพุตมาตรฐานของสั่ง cat คือชื่อไฟล์ที่อยู่ในส่วนของอาร์กิวเมนต์) เชลล์จะใช้อุปกรณ์พุตมาตรฐานที่เป็นเทอร์มินัลทันที ซึ่งนั่นก็หมายความว่าในระหว่างนี้กผู้ใช้งานพิมพ์ข้อความใด ๆ ลงไป เชลล์จะมองข้อความเหล่านั้นเป็นพุตของคำสั่ง cat ทันที เมื่อมีการป้อนข้อความไปจำนวนหนึ่งแล้วมด้วยการกดปุ่ม ENTER ซึ่งเป็นการปิดชุดข้อความบรรทัดนั้น เชลล์



รูปที่ 5 การจัดอุปกรณ์อินพุตและเอาต์พุตมาตรฐานสำหรับผู้ใช้งานทั่วไป

จะส่งกลุ่มอักขระที่ได้รับการเก็บไว้ในบัฟเฟอร์อินพุตไปยังส่วนประมวลผล ในกรณีของคำสั่ง cat ก็คือการส่งข้อความบรรทัดนั้นออกไปยังอุปกรณ์เอาต์พุตมาตรฐาน ซึ่งในที่นี้ก็คือจอภาพของเทอร์มินัลนั่นเอง ผู้ใช้งานจึงมองเห็นข้อความที่ตนเองป้อนผ่านไปแสดงกลับมาบนจอภาพ ในกรณีที่ต้องการหยุดการทำงานของโปรแกรม cat ก็ทำได้โดยการพิมพ์ CTRL-D อนึ่ง ในระหว่างการป้อนข้อความแต่ละตัวอักขระนั้น หากมีการป้อนอักขระควบคุม เช่น CTRL-H ผ่านเข้าไป เชลล์ก็จะทำการประมวลผลอักขระควบคุมนั้นทันทีดังที่ได้กล่าวมาแล้วเมื่อตอนต้น

ตัวอย่างต่อไปนี้เป็นการแสดงการโต้ตอบระหว่างผู้ใช้งานซึ่งติดต่อกับระบบปฏิบัติการ UNIX ผ่านทางอุปกรณ์อินพุตมาตรฐานที่เป็นเทอร์มินัล กับตัว UNIX เอง ซึ่งตอบกลับผ่านทางอุปกรณ์เอาต์พุตมาตรฐานที่เป็นเทอร์มินัลเช่นเดียวกัน ทั้งนี้จะใช้คำสั่ง cat เป็นคำสั่งตัวอย่างตามที่ได้กล่าวถึงข้างต้น ตัวอักษรที่พิมพ์เข้มคือสิ่งที่ผู้ใช้งานป้อนผ่านคีย์บอร์ดไป

```
$ cat
This is a test message.
This is a test message.
Pairoj is trying to present you the UNIX topics.
Pairoj is trying to present you the UNIX topics.
If you were tired typing a lot, please press CTRL-D to finish.
If you were tired typing a lot, please press CTRL-D to finish.
[CONTROL-D] <- ผู้ใช้งานกดปุ่ม CTRL พร้อมกับปุ่ม D
$
```

สำหรับเนื้อที่ของบทความในฉบับนี้ก็คงจะหมดลงเพียงเท่านี้ ฉบับหน้าผู้เขียนจะได้กล่าวต่อไปถึงรายละเอียดของเชลล์จนจบ ซึ่งจะเกี่ยวข้องกับเรื่องของการเปลี่ยนทิศทาง เรื่องของฟิลเตอร์ และอักขระพิเศษ รวมถึงเทคนิคต่าง ๆ ที่สามารถใช้กับการทำงานร่วมกับเชลล์ได้ พบกันฉบับหน้าครับ